Feature Learning by Least Generalization

Hien D. Nguyen[1,2] and Chiaki Sakama[3]

[1] University of Information Technology, Ho Chi Minh city, Vietnam
[2] Vietnam National University, Ho Chi Minh city, Vietnam
hiennd@uit.edu.vn
[3] Wakayama University, Wakayama, Japan
sakama@wakayama-u.ac.jp

Abstract. This paper provides an empirical study for feature learning based on induction. We encode image data into first-order expressions and compute their *least generalization*. An interesting question is whether the least generalization can extract a common pattern of input data. We also introduce two different methods for feature extraction based on symbolic manipulation. We perform experiments using the MNIST datasets and show that the proposed methods successfully capture features from training data and classify test data in around 90% accuracies. The results of this paper show potentials of induction and symbolic reasoning to feature learning or pattern recognition from raw data.

Keywords: Feature learning · Least generalization · Interpretable machine learning.

1 Introduction

Feature learning or *representation learning* is the technique to discover the necessary representations for feature detection or classification from data [3]. In machine learning, neural networks (NN) and *deep learning* [9] are widely used for this purpose. Deep learning has powerful capability in feature learning, while it does neither show what is learned nor explain why an output is obtained. This raises the need of *interpretable machine learning* [12] or *explainable AI* (XAI) [1] that aims to make AI systems results more understandable to humans.

Inductive logic programming (ILP) [6, 11] realizes machine learning based on symbolic reasoning. In contrast to the NN approaches, ILP can learn human-readable hypotheses from small amounts of data, which enables to accumulate learned results as knowledge and to share them by humans. One of the challenging issues of ILP is learning from raw data. In [5], the authors say:

> Most ILP systems require data in perfect symbolic form. However, much real-world data, such as images and speech, cannot easily be translated into a symbolic form. Perhaps the biggest challenge in ILP is to learn how to both perceive sensory input and learn a symbolic logic program to explain the input.

The goal of this study is to realize feature learning from raw data using ILP techniques. To this end, we first encode image data into first-order atoms by representing

pixel information in terms. Next we compute *least generalization* of those atoms. Least generalization [14, 15] is a technique of inductive generalization and extracts a common pattern among expressions. Then, an interesting question is whether least generalization can extract a common pattern of input data as features. We also introduce two different methods for feature extraction based on symbolic manipulation. We implement the proposed methods and test on the MNIST datasets. Experimental results show that some of the methods successfully extract features of handwritten digits and fashion figures in a human-readable manner. The extracted features are used for classification and their accuracy, precision and recall are evaluated.

The rest of this paper is organized as follows. Section 2 describes the proposed method, and Section 3 presents experimental results. Section 4 discusses related issues, and Section 5 summarizes the paper.

2 Feature Learning by Least Generalization

2.1 Least generalization

A *first-order language* consists of an alphabet and all formulas defined over it. The definition is the standard one in the literature [4, 6]. A *term* is either (i) a constant, (ii) a variable, or (iii) $f(t_1,\ldots,t_m)$ where f is an m-ary $(m \geq 1)$ function symbol and $t_1,\ldots,t_m$ are terms. An *atom* is a formula $P(t_1,\ldots,t_n)$ $(n \geq 1)$ where P is an n-ary predicate and t_i's are terms. An *expression* is either a term or an atom. Two atoms are *compatible* if they have the same n-ary predicate. The set of all variables (resp. terms, atoms) in the language is denoted by *Var* (resp. *Term*, *Atom*). The set of all expressions is defined as $Exp = Term \cup Atom$. A *substitution* is a mapping σ from *Var* into *Term* such that the set $\Gamma = \{\langle x, \sigma(x)\rangle \mid x \neq \sigma(x) \text{ and } x \in Var\}$ is finite. When $\sigma(x_i) = t_i$ for $i = 1,\ldots,n$, it is also written as $\sigma = \{t_1/x_1, ..., t_n/x_n\}$. The set of all substitutions in the language is denoted by *Sub*. The identity mapping ε over *Var* is the *empty substitution*.

Let $\sigma \in Sub$ and $E \in Exp$. Then $E\sigma$ is defined as follows:

$$E\sigma = \begin{cases} \sigma(x) & \text{if } E = x \text{ for } x \in Var, \\ a & \text{if } E = a \text{ for a constant } a, \\ f(t_1\sigma, ..., t_m\sigma) & \text{if } E = f(t_1, ..., t_m) \in Term, \\ P(t_1\sigma, ..., t_n\sigma) & \text{if } E = P(t_1, ..., t_n) \in Atom. \end{cases}$$

A preorder relation $\leq$ over *Atom* is defined as follows. For any $A, B \in Atom$, $A \leq B$ if $A = B\theta$ for some $\theta \in Sub$. Then, B is a *generalization* of A if $A \leq B$.

Definition 1 (least generalization). ([14, 15]) Let $A_1, A_2 \in Atom$. An atom $A \in Atom$ is a *common generalization* of A_1 and A_2 if $A_i \leq A$ for $i = 1, 2$. In particular, A is a *least common generalization* of A_1 and A_2 if A is a common generalization of A_1 and A_2, and $A \leq A'$ for any common generalization A' of A_1 and A_2. Least common generalization is simply called *least generalization*. The least generalization of A_1 and A_2 is written as $lg(\{A_1, A_2\})$.

An algorithm for computing least generalization is introduced by [14, 15]. Here we refer the one from [6] (Figure 1).[4]

[4] It is also called *anti-unification*.

Input : Two compatible atoms A_1 and A_2
Output : $G = lg(\{A_1, A_2\})$

1. Set $A'_1 = A_1$ and $A'_2 = A_2$, $\theta_1 = \theta_2 = \varepsilon$, and $i = 0$.
   Let $z_1, z_2, \ldots$ be a sequence of variables not appearing in A_1 or A_2.
2. If $A'_1 = A'_2$, then output $G := A'_1$ and stop.
3. Let p be the leftmost symbol position where A'_1 and A'_2 differ. Let s and t be the terms occurring at this position in A'_1 and A'_2, respectively.
4. If, for some j with $1 \leq j \leq i$, $z_j\theta_1 = s$ and $z_j\theta_2 = t$, then replace s at the position p in A'_1 by z_j, replace t at the position p in A'_2 by z_j, and go to 2.
5. Otherwise set i to $i+1$, replace s at the position p in A'_1 by z_i, and replace t at the position p in A'_2 by z_i. Set θ_1 to $\theta_1 \cup \{s/z_i\}$, θ_2 to $\theta_2 \cup \{t/z_i\}$, and go to 2.

Fig. 1. Algorithm for Least Generalization [6]

2.2 Encoding image data into first-order expressions

We encode image data into first-order expressions and compute their least generalization. We first describe a method of encoding image data into terms. An image (in black, white or grayscale) is presented by $28 \times 28 = 784$ pixels where each pixel is an integer value from 0 to 255.[5] An image is then represented as a vector $\boldsymbol{v} \in \mathbb{R}^{784}$ that contains pixel values as elements. Each pixel x $(0 \leq x \leq 255)$ is transformed to the term $f^k(z)$ with a variable z where

$$k = \left\lfloor \frac{x}{64} \right\rfloor + 1,$$
$$f^1(z) = f(z) \text{ and } f^{k+1}(z) = f(f^k(z)) \ \ (1 \leq k \leq 3).$$

where $\lfloor\ \rfloor$ is the floor function of a real argument x which returns the greatest integer less than or equal to x. The function symbol f is used to represent "closeness" of pixels. For instance, when $0 \leq x_1, x_2 \leq 63$, both x_1 and x_2 are represented as $f(z)$. When $x_1 = 80$ and $x_2 = 200$, for instance, x_1 is represented as $f^2(z)$ and x_2 is represented as $f^4(z)$. This representation helps to keep information of the range of shades in computing least generalization.

With this encoding, a vector $\boldsymbol{v}$ is encoded into an atom having 784 arities:

$$P(t_{1.1}, \ldots, t_{1.28}, t_{2.1}, \ldots, t_{2.28}, \ldots, t_{28.1}, \ldots, t_{28.28}).$$

In this paper, the existence of a predicate is unimportant, so hereafter we identify the above atom with the tuple $(t_{1.1}, \ldots, t_{1.28}, t_{2.1}, \ldots, t_{2.28}, \ldots, t_{28.1}, \ldots, t_{28.28})$.

[5] 0 means black and 255 means white. In between, every other number is a shade of gray ranging from black to white.

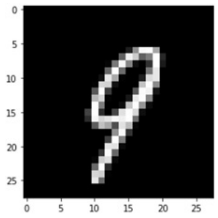

Fig. 2. An example of a 28×28 image.

Example 1. The image of Figure 2 is encoded into a tuple of terms as follows:

$$(\underbrace{f(z),\ldots,f(z),}_{28\times 6\text{ values}} \quad \%\text{1st to 6th rows}$$

$$\underbrace{f(z),\ldots,f(z),}_{16\text{ values}}\, f^3(z), f^4(z), f^4(z), f^3(z), \underbrace{f(z),\ldots,f(z),}_{8\text{ values}} \quad \%\text{7th row}$$

$$\underbrace{f(z),\ldots,f(z),}_{14\text{ values}}\, f^2(z), f^4(z), f^4(z), f^2(z), f^2(z), f^4(z), \underbrace{f(z),\ldots,f(z),}_{8\text{ values}} \quad \%\text{8th row}$$

$$\ldots \quad \underbrace{,f(z),\ldots,f(z)}_{28\text{ values}}) \quad \%\text{28th row}$$

2.3 Extracting features

Training data is classified by their labels. Suppose a set of training data $C_l = \{A_1,\ldots,A_n\}$ (called a *class*) where l is a label and A_i $(1 \leq i \leq n)$ is a first-order atom (or a tuple) representing an image. The least generalization of C_l is then computed as follows.

Algorithm 2: Least generalization of training data

Input : a set of training data $C_l = \{A_1,\ldots,A_n\}$ where A_i $(1 \leq i \leq n)$ is a first-order atom (tuple) representing an image and l is a label.
Output : least generalization of C_l (written $lg(C_l)$).

1. Put $A_0 := A_1$.
2. For i from 2 to n do:
   Compute $A_0 := lg(\{A_0, A_i\})$ by the algorithm of least generalization (Fig. 1).
3. Return A_0.

The output of Algorithm 2 is decoded into pixel data by the converse transformation of the encoding presented in Section 2.2: a term $f^k(z)$ $(1 \leq k \leq 4)$ in a tuple is converted into the pixel with the value $(k-1) \times 64$. The obtained vector $\boldsymbol{u} \in \mathbb{R}^{784}$ is viewed as features extracted by least generalization. We refer this way of extracting features by **GEN** and call $\boldsymbol{u}$ a *feature vector* by **GEN**.

Next we introduce two different methods for feature extraction. Suppose a set of training data $D_l = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ where l is a label and $\boldsymbol{v}_k \in \mathbb{R}^{784}$ $(1 \leq k \leq n)$ is a vector representing an image. Put

$$\boldsymbol{v}_k = (x^k_{1.1}, \ldots, x^k_{1.28}, x^k_{2.1}, \ldots, x^k_{2.28}, \ldots, x^k_{28.1}, \ldots, x^k_{28.28}) \qquad (1 \leq k \leq n)$$

where x^k_{ij} is a pixel value. Then, define

$$S_{ij} = \{\, x^k_{ij} \mid 1 \leq k \leq n \,\} \quad (1 \leq i, j \leq 28).$$

S_{ij} is a collection of pixel values at the location (i, j) from training data. Then, **FRQ** and **AVE** are defined as follows.

FRQ: Select the integer value u_{ij} that appears most frequently in S_{ij}.

AVE: Compute the average value v_{ij} of elements in S_{ij} and put $w_{ij} = \lfloor v_{ij} \rfloor$.

Put $\boldsymbol{u} = (u_{1.1}, \ldots, u_{1.28}, u_{2.1}, \ldots, u_{2.28}, \ldots, u_{28.1}, \ldots, u_{28.28})$ and $\boldsymbol{w} = (w_{1.1}, \ldots, w_{1.28}, w_{2.1}, \ldots, w_{2.28}, \ldots, w_{28.1}, \ldots, w_{28.28})$. Then $\boldsymbol{u}$ and $\boldsymbol{w}$ are viewed as vectors that represent features extracted by **FRQ** and **AVE**, respectively. We call $\boldsymbol{u}$ (resp. $\boldsymbol{w}$) *a feature vector* by **FRQ** (resp. **AVE**).

2.4 Classification of images

We next use the result of extracted features for classifying unlabelled test data. When there are m classes, the classification is done using the following algorithm.

Algorithm 3: Classification of test data

Input : a vector $\boldsymbol{v}$ representing an unlabelled 28×28 image, and the set of feature vectors of training data: $S = \{\, \boldsymbol{u}_k \mid k = 1, \ldots, m \,\}$.

Output : the label of $\boldsymbol{v}$.

1. For each class k $(1 \leq k \leq m)$, compute

$$D_k = \sum_{ij} | u_{ij} - v_{ij} | \qquad (1 \leq i, j \leq 28)$$

   where u_{ij} is an element in $\boldsymbol{u}_k$ and v_{ij} is an element in $\boldsymbol{v}$.
2. Return $l = k$ as the label of $\boldsymbol{v}$ where D_l is minimal among $D_1, \ldots, D_m$.

The set S of feature vectors is obtained by one of **GEN**, **FRQ**, and **AVE**. The label of a testing data is determined in a way that the sum of differences between pixel values in each location is minimal.[6]

[6] If D_l is not unique, the one with the minimum index l is selected.

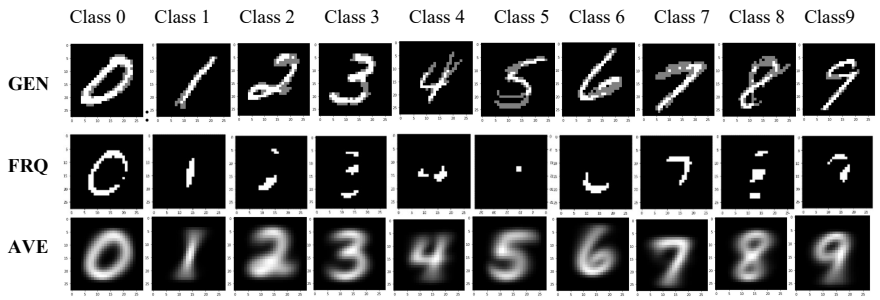

Fig. 3. Features of the MNIST dataset obtained by **GEN**, **FRQ**, and **AVE**

3 Experimental Results

We conduct experiments using two datasets, MINST hand-written digits and Fashion MNIST,[7] which are widely used as benchmark for feature learning. Each dataset is split into two parts: the training set (60,000 images) and the test set (10,000 images). The experimental testing is done by two stages: (i) extracting features from training data using **GEN**, **FRQ**, and **AVE**; and (ii) classifying test data and computing their *Precision*, *Recall*, and *Accuracy* to evaluate the methods. Precision, recall, and accuracy, which are widely used measures in machine learning, are defined as:

$$Precision := \frac{TP}{TP+FP} \quad Recall := \frac{TP}{TP+FN} \quad Accuracy := \frac{TP+TN}{TP+TN+FP+FN}$$

where TP, TN, FP, and FN mean *True Positive* (correctly predicted as positive), *True Negative* (correctly predicted as negative), *False Positive* (incorrectly predicted as positive), and *False Negative* (incorrectly predicted as negative), respectively.[8]

3.1 Testing on MNIST dataset

The MNIST data are classified into 10 classes ($0-9$). Then features are extracted in each class using **GEN**, **FRQ**, and **AVE**. The results of feature extraction is shown in Figure 3. By the figure, we observe that most digits produced by **GEN** and **AVE** are readable, while those produced by **FRQ** are less readable. The precision, recall and accuracy of three methods are summarized in Table 1. By the table, we can see that **AVE** shows the maximum average of 94% in accuracy, 80% in precision and 66% in recall. The accuracy of **GEN** and **FRQ** are around 90%. The 93% accuracy of **FRQ** is a bit surprise because the output of feature extraction is less readable. The result indicates that classification does not necessarily require the whole image of a digit.

3.2 Testing on Fashion-MNIST dataset

The Fashion-MNIST data is classified into 10 classes (Class 0:T-shirt/top, Class 1:Trouser,

[7] http://yann.lecun.com/exdb/mnist/, https://www.kaggle.com/zalando-research/fashionmnist

[8] Dataset, code, and results of testing are available at https://drive.google.com/drive/folders/1Zno76nKDhENor-lEt9sQW7dYzPlyT4JQ?usp=sharing

Table 1. The precision, recall and accuracy on the MNIST dataset

Class	**GEN**			**FRQ**			**AVE**		
	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy
0	0.42	0.91	0.94	0.75	0.65	0.94	0.82	0.94	0.98
1	0.93	0.49	0.88	0.92	0.80	0.96	0.99	0.35	0.79
2	0.19	0.86	0.91	0.39	0.65	0.92	0.42	0.97	0.94
3	0.61	0.50	0.90	0.70	0.61	0.92	0.62	0.72	0.94
4	0.80	0.34	0.83	0.79	0.71	0.95	0.67	0.83	0.96
5	0.51	0.38	0.88	0.24	0.34	0.89	0.33	0.87	0.94
6	0.44	0.60	0.92	0.72	0.67	0.94	0.79	0.86	0.97
7	0.17	0.61	0.90	0.70	0.62	0.92	0.77	0.86	0.96
8	0.22	0.66	0.91	0.59	0.59	0.92	0.44	0.86	0.94
9	0.33	0.32	0.86	0.52	0.58	0.91	0.78	0.69	0.94
Average	**0.46**	**0.57**	**0.89**	**0.63**	**0.62**	**0.93**	**0.66**	**0.80**	**0.94**

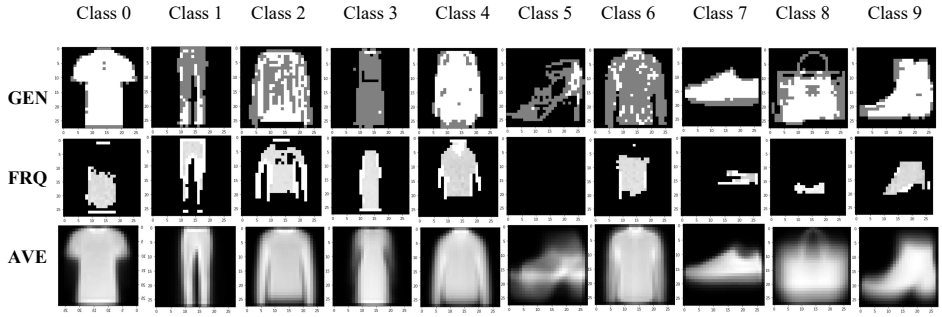

Fig. 4. Features of the Fashion-MNIST dataset obtained by **GEN**, **FRQ**, and **AVE**

Class 2:Pullover, Class 3:Dress, Class 4:Coat, Class 5:Sandal, Class 6: Shirt, Class 7: Sneaker, Class 8: Bag, Class 9:Ankle boot). Figure 4 shows the results of feature extraction using three methods. By the figure, we can observe that **GEN** and **AVE** capture the shape of each class with a few exceptions. Again, the output of **FRQ** is less clearer than the others. The precision, recall and accuracy of each method are computed using test data as in Table 2. As before, **AVE** outputs the highest values among three methods. It is known that Fashion-MNIST is significantly harder than MNIST, while the proposed three methods still keeps around the 90% accuracy.

4 Discussion

Feature learning from labelled data has been done using neural networks (NN). In deep leaning models, features are extracted in hidden layers and then represented by the neurons of the network. However, what is learned in NN is uninterpretable and left as a black box. In contrast to the NN approaches, our approach based on symbolic reasoning is interpretable and transparent. Representing an image data as a vector with

Table 2. The precision, recall and accuracy on the Fashion-MNIST dataset

Class	**GEN**			**FRQ**			**AVE**		
	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy
0	0.61	0.48	0.90	0.16	0.23	0.86	0.62	0.76	0.94
1	0.87	0.72	0.95	0.83	0.97	0.98	0.94	0.76	0.96
2	0.21	0.47	0.90	0.44	0.40	0.88	0.25	0.64	0.91
3	0.25	0.22	0.84	0.48	0.82	0.94	0.69	0.55	0.91
4	0.68	0.37	0.85	0.63	0.27	0.80	0.66	0.43	0.88
5	0.43	0.73	0.93	0.56	0.32	0.84	0.53	0.42	0.88
6	0.14	0.35	0.89	0.01	0.06	0.89	0.17	0.37	0.89
7	0.89	0.54	0.91	0.37	0.74	0.92	0.89	0.57	0.92
8	0.47	0.96	0.95	0.046	0.155	0.88	0.56	0.96	0.95
9	0.80	0.80	0.96	0.93	0.52	0.91	0.79	0.85	0.97
Average	**0.54**	**0.56**	**0.91**	**0.45**	**0.45**	**0.89**	**0.61**	**0.63**	**0.92**

pixel values, **GEN**, **FRQ**, and **AVE** compute feature vectors that represent features of training data. Functions computing those features are explicitly given, then one can understand reasons why test data is classified into some classes. In this respect, feature learning in this paper realizes *interpretable machine learning*. Given n training data, **GEN** is computed in $O(n \times \log^2 m)$ using m processors [13], and **FRQ** and **AVE** are computed in $O(n)$. Classification of a test data is done in $O(l \times k^2)$ where l is the number of elements in a vector $\boldsymbol{v}$ and k is the number of classes.

There are few studies on encoding image data into first-order formulas except [2], which introduces an NN architecture called the *first-order state autoencoder* (FOSAE). Given the feature vectors of objects in the environment, FOSAE automatically learns to identify a set of predicates (relations) as well as to select appropriate objects as arguments for the predicates. The resulting representation is used for classical planning. The goal of [2] is not feature learning but predicate symbol grounding. It does not use induction but uses NN to detect common pattern between objects that define a relation.

There are some approaches for integrating low-level perception in NN with high-level reasoning in LP. *Differentiable ILP* (∂ILP) [8] combines ILP and NN and learns symbolic rules that are robust to noisy and ambiguous data. In the MNIST classification task, a convolutional NN (ConvNet) is connected to ∂ILP. When an image is fed into the pretrained ConvNet, it predicts a probability distribution for the target variable. The image is then converted to the most probable atom and is merged to ILP. *DeepProbLog* [10] integrates NN and probabilistic LP in a way that the output of NN is encapsulated in the form of *neural predicates*. In *abductive learning* [7], NN is used for obtaining pseudo-labels from training data, which are then treated as groundings of the primitive concepts for abductive reasoning in LP. These studies combine background knowledge represented as LP with the output of NN, while raw data is processed using NN.

5 Concluding Remarks

This paper introduced new methods that learn features of labelled images by symbolic reasoning, then classify unlabelled images by comparing them with learned feature vec-

tors. Our approach is purely symbolic and does not use NN for learning from image data. To the best of our knowledge, this is the first attempt that realizes feature learning using symbolic reasoning without relying on NN. Although the classification accuracy achieved in this paper is still inferior to the state of the art of NN technologies,[9] the current paper shows potentials of symbolic reasoning for feature learning from raw data.

For efficient implementation, an algorithm proposed in [13] realizes parallel computation of least generalization. In this paper, a pixel x is transformed to a term $f^k(z)$ in a way that black is represented by $f(z)$ and white is represented by $f^4(z)$. This means that least generalization of black and white becomes black, while one may argue that the result should be gray. It would be interesting to realize an alternative transformation from pixel values to terms and compare their effects. We continue to investigate more robust and effective representation of raw data in terms of symbolic expressions, and its application to data other than images.

References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE Access 6: 52138–52160 (2018)
2. Asai, M.: Unsupervised grounding of plannable first-order logic representation from images. in: Proc. 21th Int'l Conf. Automated Planning and Scheduling, pp. 583–591 (2019)
3. Bengio, Y., Courville, A.: Representation learning: a review and new perspectives. IEEE Trans. Pattern Analysis and Machine Intelligence 35(8): 1798–1828 (2013)
4. Chang, C. L., Lee, R. T. C.: Symbolic Logic and Mechanical Theorem Proving. Academic Press, New York (1973)
5. Cropper, A., Dunmančić, S.: Inductice logic programming 30: a new introduction. arXiv:2008.07912v3 (2021)
6. N-Cheng, S-H., De Wolf, R. Foundations of Inductive Logic Programming. LNAI, vol. 1228, Springer, Berlin, Heidelberg (1997)
7. Dai, W., Xu, Q., Yu, Y., Zhou, Z. Bridging machine learning and logical reasoning by abductive learning. Advances in Neural Info. Processing Systems 32, pp. 2811–2822 (2019)
8. Evans, R., Grefenstette, E.: Learning explanatory rules from noisy data. Journal of AI Research 61:1–64 (2018)
9. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521, pp. 436–444 (2015)
10. Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., De Raedt, L. DeepProbLog: Neural probabilistic logic programming. In: Advances in Neural Information Processing Systems 31, pp. 3753–3763 (2018)
11. Muggleton (ed.): Inductive Logic Programming, Academic Press (1992)
12. Molnar, C.: Interpretable Machine Learning: A Guide for Making Black Box Models Explainable, Lulu.com (2020)
13. Nguyen, H. D., Sakama, C.: A new algorithm for computing least generalization of a set of atoms, in: Proc. 29th Int'l Conf. Inductive Logic Programming, LNAI, vol. 11770, pp. 81–97, Springer, Berlin, Heidelberg (2019)
14. Plotkin, G. D.: A note on inductive generalization. Machine Intelligence, vol. 5, Edinburgh University Press, pp. 153–163 (1970)
15. Reynolds, J. C.: Transformational systems and the algebraic structure of atomic formulas. Machine Intelligence, vol. 5, Edinburgh Univ. Press, pp. 135–151 (1970)

[9] 99.87% accuracy (9.2021). https://paperswithcode.com/sota/image-classification-on-mnist